

Semestrálna práca MI-PAP 2013/2014:

Násobenie matíc

Pavol Kušlita

Martin Klepáč

April 1, 2014

1 Definícia problému

Našou úlohou bolo vytvoriť program, ktorý implementuje násobenie matíc formou ako klasického algoritmu, tak aj s použitím Strassenovho algoritmu.

2 Formát vstupu, výstupu

Formálne, vstup vyjadríme pomocou

- A, B = vstupné matice
- A_x, A_y, B_x, B_y = dimenzie vstupných matíc A, B

Výstupom algoritmu je matica C s dimenziami A_x, B_y za podmienky, že $A_y = B_x$. V opačnom prípade nie je možno vykonať násobenie vstupných matíc.

3 Implementácia sekvenčného riešenia

Sekvenčný algoritmu pozostáva z trojice cyklov, ktoré prechádzajú matice A, B v nasledovnom poradí:

1. riadok matice A
2. stĺpec matice B
3. stĺpec matice A , ktorý zároveň predstavuje riadok matice B

Vo najvnútornejšom cykle sa potom vykoná samotný výpočet popísaný pseudokódom nižšie:

```
C[i][j] += A[i][k] * B[k][j];
```

Zložitosť takéhoto výpočtu je potom intuitívne $O(n^3)$. Zároveň tento triviálny algoritmus nevyužíva možnosti cache pamäte.

Vylepšenie tohto triviálneho algoritmu spočíva v použití techniky loop-tiling. Výsledkom je zdvojnásobenie celkového počtu cyklov na 6, pričom miesto sekvenčného posunu indexov i až k posúvame indexy v našom prípade o 100, aby sme následne iterovali pomocnými indexami ii , jj respektíve kk medzi 0-99, 100-199 a tak ďalej. Výsledkom je rovnaké množstvo vykonanej práce pri efektívnejšom využití cache pamäte. Pseudokód popisujúci techniku loop tiling-u:

```
for i in 0..Ax
  for j in 0..By
    for k in 0..Ay
      for ii in i..i+100
        for jj in j..j+100
          for kk in k..k+100
            C[ii][jj] += A[ii][kk] * B[kk][jj];
```

<TO DO - Strassenov algoritmus sekvenčne>

4 Implementácia paralelného riešenia pomocou OpenMP

Prostredie OpenMP sa ukázalo byť veľmi jednoduché na implementáciu triviálneho algoritmu s použitím viacerých vlákien. Vzhľadom na to, že objem práce v najvnútornejšom cykle je rovnaký pre všetky vlákna, použili sme statické rozdelenie záťaže priamo v prostredí OpenMP definované kľúčovým slovom *static*. Aby sme v prípade menších vstupných matíc v kombinácii s väčším počtom vlákien zamedzili plytvaniu prostriedkov, miesto jedného cyklu paralelizujeme dvojicu vonkajších cyklov s pomocou kľúčového slova *collapse*.

Kód definujúci paralelizáciu ako triviálneho algoritmu, tak aj algoritmu s použitím loop tiling-u vyzerá nasledovne:

```
#pragma omp parallel for collapse (2) default(shared) private(i,j,k)
schedule(static)
```

<TO DO - Strassenov algoritmus paralelne>

Pre potreby merania na serveri star.fit.cvut.cz sme vygenerovali trojicu pseudonáhodných štvorcových matíc o veľkosti 1024, 2048 a 4096. Program samozrejme dokáže spracovať aj iné ako štvorcové matice, ale pre férové porovnanie Strassenovho algoritmu s ostatnými riešeniami, používame práve štvorcové matice o veľkosti 2^n - naša implementácia Strassena automaticky dopĺňa vstupné matice na najbližšiu vyššiu takúto veľkosť.

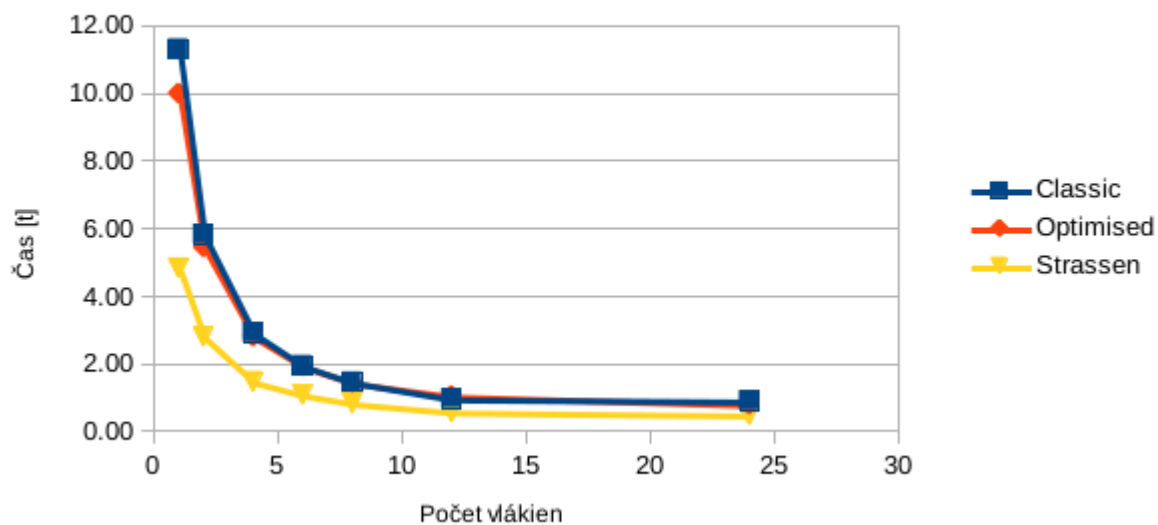
Namerané hodnoty sú uvedené v tabuľke 1 pre dané vstupné matice, počet vlákien 1, 2, 4, 6, 8, 12, 24 pre všetky hore popísané implementácie. Výsledné hodnoty predstavujú aritmetický priemer trojice meraní.

Nasledovné grafy znázorňujú ako paralelný čas, tak aj paralelnú cenu pre všetky tri vstupy pre variabilný počet vlákien.

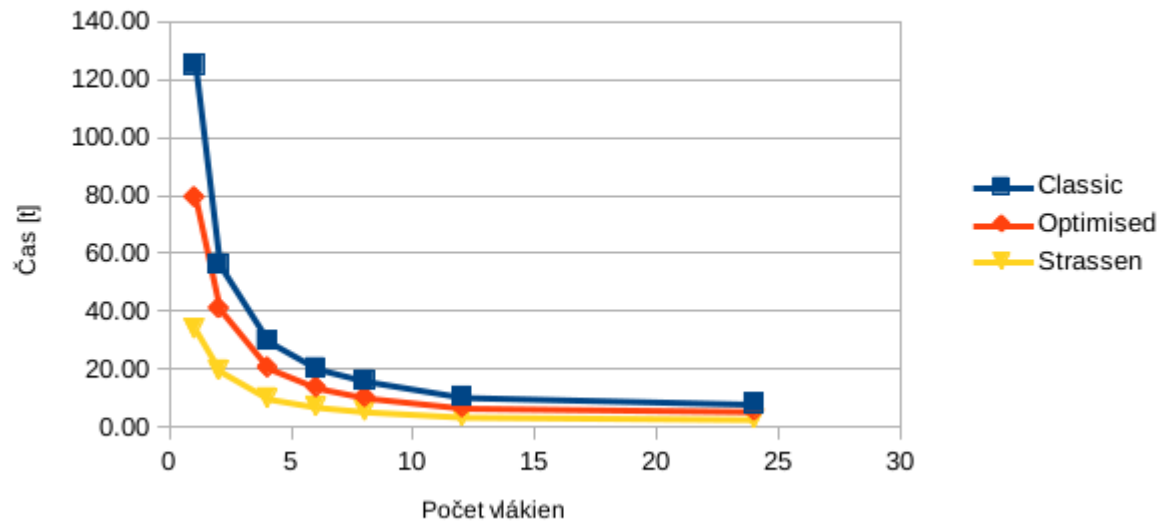
	1024x1024			2048x2048			4096x4096		
	CL	LT	ST	CL	LT	ST	CL	LT	ST
-n 1	11.33	10.02	4.85	125.20	79.60	34.15	1347.2	651.67	239.81
-n 2	5.84	5.47	2.82	56.44	41.22	19.82	565.86	327.14	138.82
-n 4	2.93	2.82	1.49	30.03	20.79	10.21	302.63	163.87	70.66
-n 6	1.96	1.94	1.10	20.59	13.96	7.35	203.99	109.19	52.04
-n 8	1.48	1.46	0.85	16.33	10.57	5.80	158.66	81.65	39.93
-n 12	0.98	1.07	0.60	10.74	7.05	3.96	114.60	56.16	27.91
-n 24	0.92	0.81	0.50	8.40	5.87	3.14	83.37	50.18	19.71

Table 1: Trvanie behu v sekundách na OpenMP (CL = classic, LT = loop-tiling, ST = Strassen)

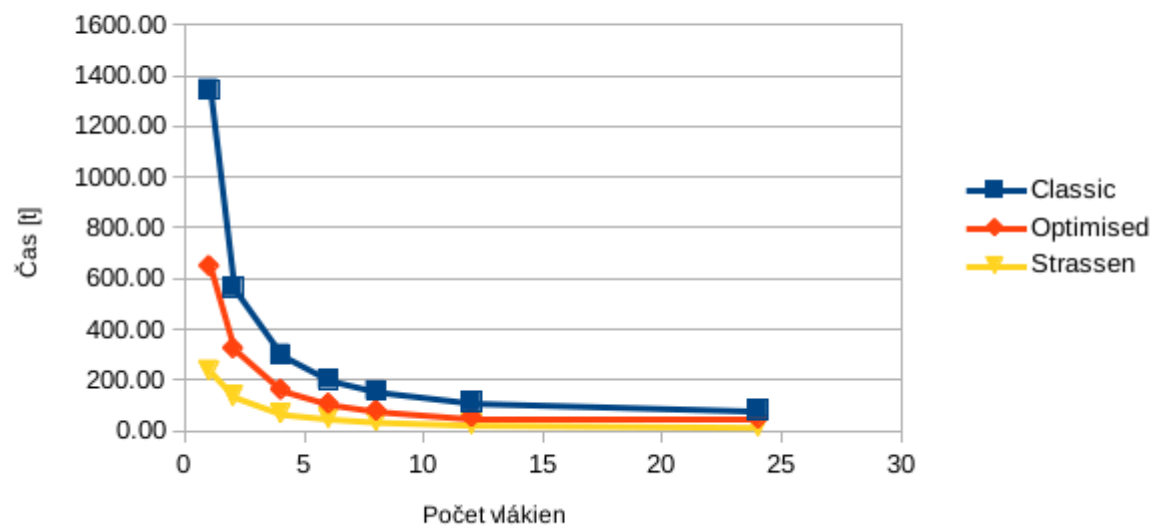
Výpočet na OpenMP, vstup 1024x1024



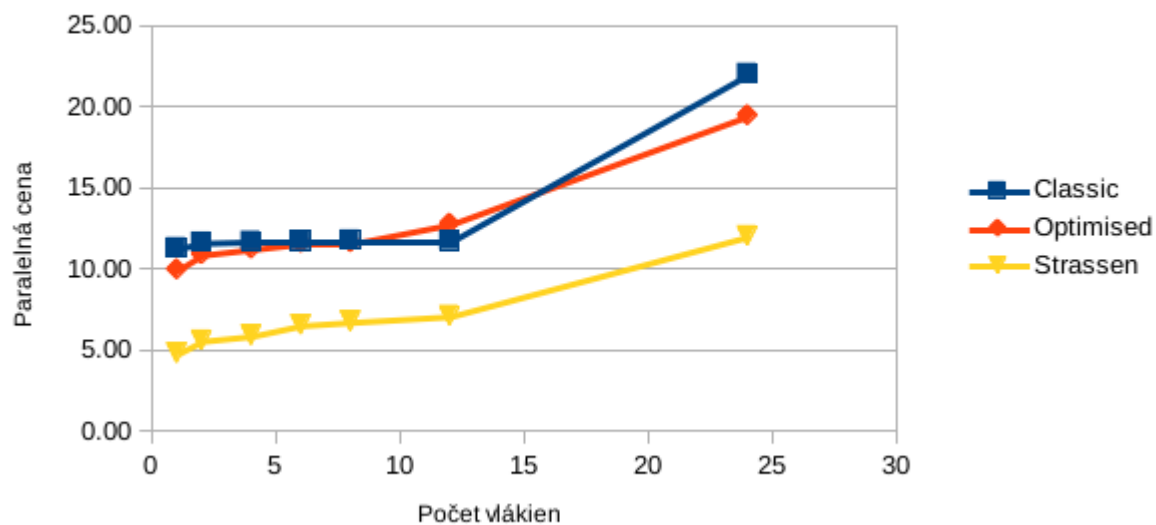
Výpočet na OpenMP, vstup 2048x2048



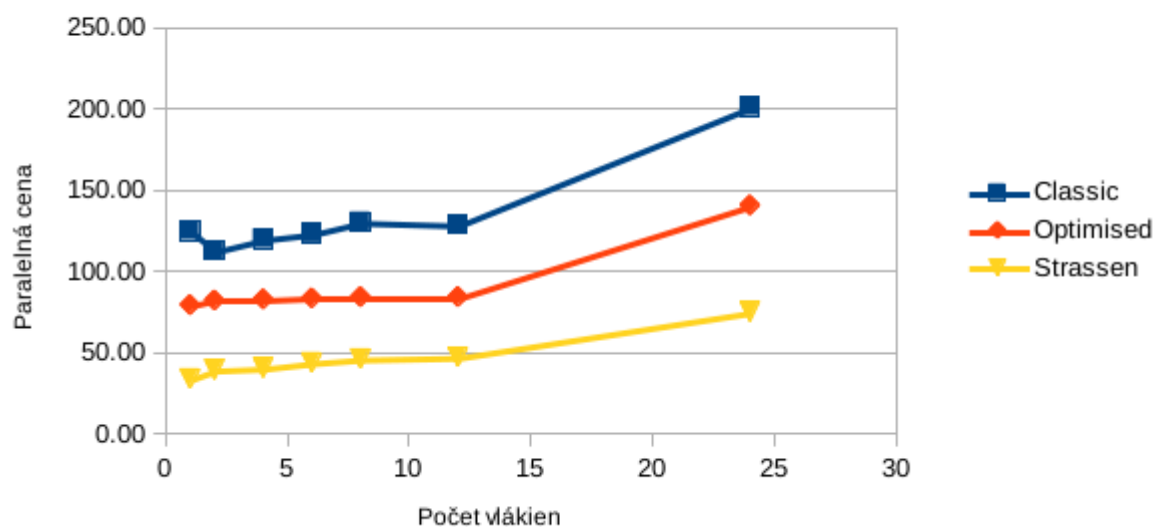
Výpočet na OpenMP, vstup 4096x4096



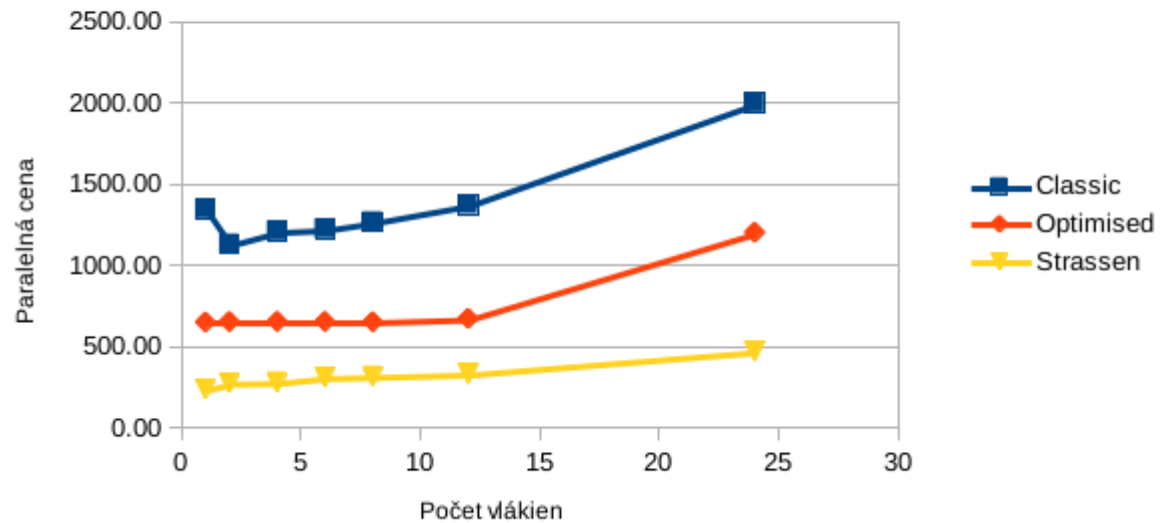
Paralelná cena na OpenMP, vstup 1024x1024



Paralelná cena na OpenMP, vstup 2048x2048



Paralelná cena na OpenMP, vstup 4096x4096



Ako môžeme vidieť, pre takmer všetky vlákna (výnimkou je 24 vlákien) a všetky implementácie sme dosiahli približne lineárne zrýchlenie, a teda konštantnú paralelnú cenu. Zvýšenie paralelnej ceny pre 24 vlákien spočíva v tom, že nami použitý hardvér nemal k dispozícii 24 fyzických procesorov, a teda zákonite došlo k prepínaniu kontextu v rámci jedného fyzického procesora. V porovnaní s predmetom BI-PPR, na ktorom sme pracovali s distribuovanou pamäťou, nedošlo k superlineárnemu zrýchleniu.

Ak by sme porovnávali jednotlivé implementácie medzi sebou, klasický a optimalizovaný algoritmus majú rovnakú asymptotickú zložitosť. Napriek tomu dosahuje optimalizovaná verzia zrýchlenie približne v rozsahu 10 až 50 percent v závislosti od veľkosti vstupu (čím väčší vstup, tým väčšie zrýchlenie). Strassenov algoritmus, ktorého zložitosť je $O(n^{2.8074})$, dosahuje v porovnaní s optimalizovanou variantou zrýchlenie približne o 50 percent.